facebook
Artificial Intelligence Research

How to solve an MDP incrementally: Reinforcement Learning

Matteo Pirotta
Facebook AI Research

Acknowledgments

Special thanks to Alessandro Lazaric for providing these slides from the RL class we teach in Paris.

The value functions can be represented **exactly** (e.g., tabular setting).

Learning From Samples

- Dynamic programming algorithms require an *explicit* definition of
  - transition probabilities $p(\cdot|s,a)$
  - reward function $r(s,a)$

Learning From Samples

- Dynamic programming algorithms require an *explicit* definition of
  - transition probabilities $p(\cdot|s,a)$
  - reward function $r(s,a)$
- This knowledge is often *unavailable* (i.e., wind intensity, human-computer-interaction).

Learning From Samples

- Dynamic programming algorithms require an *explicit* definition of
  - transition probabilities $p(\cdot|s,a)$
  - reward function $r(s,a)$
- This knowledge is often *unavailable* (i.e., wind intensity, human-computer-interaction).
- *Can we relax this assumption?*

Setting

- *Learning with generative model.* A *black-box simulator* f of the environment is available. Given (s, a),

$$f(s, a) = \{s', r\} \text{ with } s' \sim p(\cdot|s, a), r = r(s, a).$$

Setting

- *Learning with generative model.* A *black-box simulator* f of the environment is available. Given (s, a),

$$f(s, a) = \{s', r\} \text{ with } s' \sim p(\cdot|s, a), r = r(s, a).$$

- *Episodic learning.* Multiple *trajectories* can be repeatedly generated from some initial states and terminating when a *reset* condition is achieved:

$$\left(s_{0,i}, s_{1,i}, \ldots, s_{T_i,i}\right)_{i=1}^{n}$$

Setting

- *Learning with generative model.* A *black-box simulator* f of the environment is available. Given (s, a),

$$f(s, a) = \{s', r\} \text{ with } s' \sim p(\cdot|s, a), r = r(s, a).$$

- *Episodic learning.* Multiple *trajectories* can be repeatedly generated from some initial states and terminating when a *reset* condition is achieved:

$$\left(s_{0,i}, s_{1,i}, \ldots, s_{T_i,i}\right)_{i=1}^{n}$$

- *Online learning.* At each time t the agent is at state s_t, it takes action a_t, it observes a transition to state s_{t+1}, and it receives a reward r_t. We *assume* that $s_{t+1} \sim p(\cdot|s_t, a_t)$ and $r_t = r(s_t, a_t)$ (i.e., MDP assumption). No *reset*.

RL Interaction Protocol

Let $\mathcal{F}_t$ be the filtration generated up to time t

$$\mathcal{F}_t = \sigma(s_0, a_0, r_0, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$$

At each time t

- select an action $a_t \sim \pi(s_t)$

$$\pi_t \text{ is } \mathcal{F}_t - \text{measurable}$$

- observe the reward and next state

$$r_t \sim \nu(s_t, a_t), \quad s_{t+1} \sim p(\cdot|s_t, a_t)$$

Reinforcement Learning Algorithms

Reinforcement Learning Algorithms

Policy Evaluation

Policy Learning

Outline

Policy Evaluation

Fixed policy π

For $i = 1, \ldots, n$

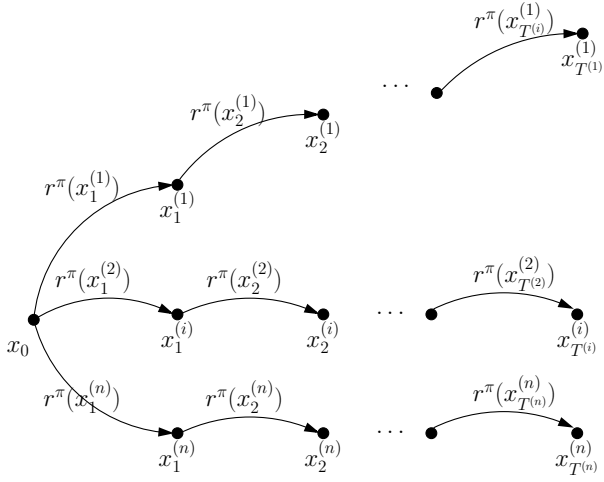
1. Set $t = 0$
2. Set initial state s_0
3. **While** ($s_{t,i}$ not terminal) *[execute one trajectory]*
   1. Take action $a_{t,i} = \pi(s_{t,i})$
   2. Observe **next state** $s_{t+1,i}$ and **reward** $r_{t,i} = r(s_{t,i}, a_{t,i}) \sim \nu(s_{t,i}, a_{t,i})$
   3. Set $t = t + 1$

   EndWhile

EndFor

Return: Estimate of the value function $\widehat{V}^{\pi}(\cdot)$

The RL Interaction Protocol



Monte-Carlo Estimation

Cumulative sum of rewards

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) | s_0 = s; \pi\right]$$

- Given n trajectories under π starting from $s_{0,i} = s$

$$h^i = (s_{0,i}, a_{0,i}, r_{0,i}, \ldots, s_{T_i,i})$$

we can estimate the value function as

$$\widehat{V}_n^{\pi}(s_0) = \frac{1}{n}\sum_{i=1}^{n}\Bigg[\underbrace{\sum_{t=0}^{T_i}\gamma^t r_{t,i}}_{:=\widehat{R}_i(s_0)}\Bigg]$$

Monte-Carlo Approximation: Properties

Theorem

The returns used in the Monte-Carlo estimation starting from an initial state s_0 are unbiased estimators of V^π

$$\mathbb{E}[\widehat{R}_i(s_0)] = \mathbb{E}\Big[r_0 + \gamma r_{1,i} + \cdots + \gamma^{T_i} r_{T_i,i}\Big] = V^\pi(s_0).$$

Furthermore, the Monte-Carlo estimator converges to the value function

$$\widehat{V}_n^\pi(s_0) \xrightarrow{a.s.} V^\pi(s_0).$$

- It applies to any state s used as the *beginning* of a trajectory (sub-trajectories could be used in practice)
- Finite-sample guarantees are possible (after n trajectories)

Monte-Carlo Approximation: Extensions

Non-episodic problems

- Interrupt trajectories after H steps

$$\widehat{R}_i(s_0) = \sum_{t=0}^{H} \gamma^t r_{t,i}$$

- Every return is ignoring a term $\sum_{t=H+1}^{\infty} \gamma^t r_{t,i}$

Monte-Carlo Approximation: Properties

Theorem

*The Monte-Carlo estimator computed over H steps converges to a **biased** value function*

$$\widehat{V}_n^{\pi}(s_0) \xrightarrow{a.s.} \overline{V}_H^{\pi}(s_0),$$

such that

$$\left|\overline{V}_H^{\pi}(s_0) - V^{\pi}(s_0)\right| \leq \gamma^H \frac{r_{\max}}{1-\gamma}$$

Monte-Carlo: an Incremental Implementation

General Monte Carlo estimate. Let Z_i be i.i.d. with mean μ, then

$$\widehat{\mu}_n = \frac{1}{n}\sum_{i=1}^{n} Z_i$$

Incremental update

$$\begin{aligned}\widehat{\mu}_n &= \frac{n-1}{n}\widehat{\mu}_{n-1} + \frac{1}{n}Z_n \\ &= \widehat{\mu}_{n-1} + \alpha(n)(Z_n - \widehat{\mu}_{n-1})\end{aligned}$$

with $\alpha(n) = \frac{1}{n}$

Can we use different learning rates?

Stochastic Approximation

Goal : Find the solution to $\phi(x^\star) = 0$ based on access to noisy function evaluations, i.e. for every x, one can observe a random value

$$Y = \phi(x) + \epsilon$$

where ϵ has a zero mean (conditionally to previous queries)

Stochastic Approximation

Robbins-Monro algorithm

Given an initial x_0, for all $n \geq 1$

- observe $Y_n = \phi(x_{n-1}) + \epsilon_n$
- update $x_n = x_{n-1} + \alpha_n Y_n$

☞ Special case: estimate a mean μ based on iid samples Z_i

$$\phi(x) = \mu - x$$

with $Y_n = Z_n - \widehat{\mu}_{n-1}$, the RM update is indeed

$$\widehat{\mu}_n = \widehat{\mu}_{n-1} + \alpha_n (Z_n - \widehat{\mu}_{n-1})$$

Stochastic Approximation

Robbins-Monro Convergence

Let $\phi : \mathcal{I} \subseteq \mathbb{R} \to \mathbb{R}$. Under the following assumptions

- ϕ is continuous and $\forall x \neq x^\star$, $(x - x^\star)\phi(x) < 0$
- there exists $C > 0$ such that $\mathbb{E}[Y_n | x_{n-1}] \leq C(1 + x_{n-1}^2)$
- the sequence $\{\alpha_n\}$ satisfy

$$\sum_{n=1}^{\infty} \alpha_n = \infty \quad \text{and} \quad \sum_{n=1}^{\infty} \alpha_n^2 < \infty \tag{1}$$

the Robbins-Monro algorithm is such that $x_n \xrightarrow{a.s.} x^\star$

☞ the mean estimator converges to μ for any sequence $\{\alpha_n\}$ satisfying (1)

Monte-Carlo for Value Function

Estimated value function

$$\widehat{V}_n^{\pi}(s_0) = \frac{1}{n}\sum_{i=1}^{n}\widehat{R}_i(s_0) = \frac{n-1}{n}\widehat{V}_{n-1}^{\pi}(s_0) + \frac{1}{n}\widehat{R}_n(s_0)$$
$$\approx \big(1-\alpha(n)\big)\widehat{V}_{n-1}^{\pi}(s_0) + \alpha(n)\widehat{R}_n(s_0)$$

- Incremental Monte-Carlo estimation converges to V^{π}

$$\widehat{V}_n^{\pi}(s_0) \xrightarrow{a.s.} V^{\pi}(s_0).$$

  for a wide range of choice of learning rate schemes.
- This scheme is often referred to as *TD(1)*.

Outline

Robbins-Monro for fixed points

Goal : Find the solution to $x^\star = T(x^\star)$ based on access to noisy evaluations of $T(x)$.

Stochastic approximation for a fixed point

Given an initial x_0, for all $n \geq 1$

- a noisy evaluation Z_n: $\mathbb{E}[Z_n | x_{n-1}] = T(x_{n-1})$
- update $x_n = x_{n-1} + \alpha_n (Z_n - x_{n-1})$

👉 corresponds to the Robbins-Monro algorithm with

$$\phi(x) = T(x) - x \quad \text{and} \quad Y_n = Z_n - x_{n-1}$$

State Value Function

Fixed point of Bellman equation

$$V^{\pi}(s) = r^{\pi}(s) + \gamma \sum_{s'} p^{\pi}(s'|s) V^{\pi}(s')$$

Temporal-Difference TD(0) Estimation

At each step t, observe s_t, r_t, s_{t+1} following π and define

$$Z_t = r_t + \gamma \widehat{V}^{\pi}(s_{t+1}) \qquad [\text{noisy evaluations of } \mathcal{T}^{\pi}\widehat{V}(s_t)]$$

Then, the "Robbins-Monro" update is

$$\begin{aligned}\widehat{V}^{\pi}(s_t) &= \widehat{V}^{\pi}(s_t) + \alpha\Big(Z_t - \widehat{V}^{\pi}(s_t)\Big) \\ &= (1-\alpha)\widehat{V}^{\pi}(s_t) + \alpha Z_t\end{aligned}$$

Temporal-Difference TD(0) Estimation

Interpretation: temporal-difference error

The Robbins-Monro update rewrites

$$\widehat{V}^{\pi}(s_t) = \widehat{V}^{\pi}(s_t) + \alpha_t \delta_t$$

defining the **Temporal difference error** of estimate $\widehat{V}^{\pi}$ w.r.t. transition (s_t, r_t, s_{t+1}):

$$\delta_t = r_t + \gamma \widehat{V}^{\pi}(s_{t+1}) - \widehat{V}^{\pi}(s_t)$$

- Bellman error for function $\widehat{V}$ at state s

$$\mathcal{B}^{\pi}(\widehat{V}; s) = \mathcal{T}^{\pi}\widehat{V}(s) - \widehat{V}(s) = r^{\pi}(s) + \gamma \sum_{s'} p^{\pi}(s'|s)\widehat{V}(s') - \widehat{V}(s) \qquad [\mathcal{B}^{\pi}(V^{\pi}; s) = 0]$$

- Conditioned on s_t, δ_t is an *unbiased* estimate of $\mathcal{B}^{\pi}$

$$\mathbb{E}_{r_t, s_{t+1}}\left[\delta_t \middle| s_t\right] = r^{\pi}(s_t) + \gamma \mathbb{E}_{s_{t+1}|s_t}\left[\widehat{V}^{\pi}(s_{t+1})\right] - \widehat{V}^{\pi}(s_t) = \mathcal{B}^{\pi}(\widehat{V}^{\pi}, s_t)$$

- Expected dynamics of TD(0)

$$\overline{V}^{\pi}(s_t) = \overline{V}^{\pi}(s_t) + \alpha(s_t)\mathcal{B}^{\pi}(\overline{V}^{\pi}; s_t)$$

Temporal-Difference TD(0) Estimation

Interpretation: moving average

- Mix between old and new estimate of $V^{\pi}(s_t)$

  old estimate $\widehat{V}^{\pi}(s_t)$ new estimate $r_t + \gamma \widehat{V}^{\pi}(s_{t+1})$

- **Weighted average**

$$\widehat{V}^{\pi}(s_t) = (1-\alpha)\widehat{V}^{\pi}(s_t) + \alpha\Big(r_t + \gamma \widehat{V}^{\pi}(s_{t+1})\Big)$$

Temporal Difference $TD(0)$

Input: π, T, s_0, V_0
$V = V_0$, $s = s_0$
$N = 0_S$
for $t = 1, \ldots, T$ **do**
  N(s) = N(s) + 1
  Execute action $a \sim \pi(s)$
  Observe r and next state s'
  Update

$$\widehat{V}(s) = \widehat{V}(s) + \alpha_{N(s)}(r + \gamma \widehat{V}(s') - \widehat{V}(s))$$

  $s = s'$
end
return V

Possible variants

- instead of a very long trajectory, smaller trajectories with restarts
- update V for all states only at the end of each trajectory (offline)
- for each state s, update V(s) at most once per trajectory (first visit)

Temporal-Difference TD(0): Properties

Theorem

Let TD(0) run with learning rate $\alpha(N_t(s_t))$ where is the number of visits to the state s_t. If all states are visited infinitely often and the learning rate is set such that

$$\sum_{t=0}^{\infty} \alpha(t) = \infty \qquad \sum_{t=0}^{\infty} \alpha(t)^2 < \infty \qquad \textit{[Robbins Monro's condition]}$$

then for any state $s \in \mathcal{S}$

$$\widehat{V}^{\pi}(s) \xrightarrow{a.s.} V^{\pi}(s).$$

☞ standard choise is $\alpha_t(s) = \frac{1}{i^{\beta}}$ for $\beta \in (1/2, 1]$

$$\widehat{V}_t(s) = \widehat{V}_{t-1}(s) + \frac{1}{N_t(s)^{\beta}}(r + \gamma \widehat{V}_{t-1}(s') - \widehat{V}_{t-1}(s))$$

where $N_t(s)$ is the number of visits to state s before t

From Monte-Carlo to Temporal Differences

Temporal difference $\delta_t = r_t + \gamma \widehat{V}^{\pi}(s_{t+1}) - \widehat{V}^{\pi}(s_t)$

Update after the n-th trajectory

$$
\begin{aligned}
\widehat{V}^{\pi}_{n+1}(s_0) &= (1-\alpha_{n+1})\widehat{V}^{\pi}_{n}(s_0) + \alpha_{n+1}\widehat{R}_{n+1}(s_0) \\
&= \widehat{V}^{\pi}_{n}(s_0) + \alpha_{n+1}\Big(\widehat{R}_{n+1}(s_0) - \widehat{V}^{\pi}_{n}(s_0)\Big) \quad \text{[MC]} \\
&= \widehat{V}^{\pi}_{n}(s_0) + \alpha_{n+1}\Big(r_{0,n} + \gamma r_{1,n} + \gamma^2 r_{2,n} + \gamma^3 r_{3,n} + \cdots - \widehat{V}^{\pi}_{n}(s_0)\Big) \\
&= \widehat{V}^{\pi}_{n}(s_0) + \alpha_{n+1}\Big(r_{0,n} + \gamma\widehat{V}_{n}(s_{1,n}) - \widehat{V}^{\pi}_{n}(s_0) - \gamma\widehat{V}^{\pi}_{n}(s_{1,n}) + \gamma r_{1,n} + \gamma^2 r_{2,n} + \gamma^3 r_{3,n} + \cdots\Big) \\
&= \widehat{V}^{\pi}_{n}(s_0) + \alpha_{n+1}\Big(\delta_{0,n} - \gamma\widehat{V}^{\pi}_{n}(s_{1,n}) + \gamma r_{1,n} + \gamma^2 r_{2,n} + \gamma^3 r_{3,n} + \cdots\Big) \\
&= \widehat{V}^{\pi}_{n}(s_0) + \alpha_{n+1}\Big(\delta_{0,n} + \gamma r_{1,n} + \gamma^2\widehat{V}^{\pi}_{n}(s_{2,n}) - \gamma\widehat{V}^{\pi}_{n}(s_{1,n}) - \gamma^2\widehat{V}^{\pi}_{n}(s_{2,n}) + \gamma^2 r_{2,n} + \gamma^3 r_{3,n} + \cdot \\
&= \widehat{V}^{\pi}_{n}(s_0) + \alpha_{n+1}\Big(\delta_{0,n} + \gamma\delta_{1,n} - \gamma^2\widehat{V}^{\pi}_{n}(s_{2,n}) + \gamma^2 r_{2,n} + \gamma^3 r_{3,n} + \cdots\Big) \\
&= \widehat{V}^{\pi}_{n}(s_0) + \alpha_{n+1}\Big(\delta_{0,n} + \gamma\delta_{1,n} + \gamma^2\delta_{2,n} + \ldots + \gamma^{T_n-1}\delta_{T_n,n}\Big)
\end{aligned}
$$

Comparison between Inc-MC and TD(0)

Temporal difference $\delta_t = r_t + \gamma \widehat{V}^\pi(s_{t+1}) - \widehat{V}^\pi(s_t)$

- Incremental Monte-Carlo

$$\widehat{V}^\pi(s_0) = \widehat{V}^\pi(s_0) + \alpha[\delta_0 + \gamma\delta_1 + \cdots + \gamma^{T-1}\delta_T].$$

$\Rightarrow$ *No bias*, *large variance* *[long trajectory]*

TD(0)

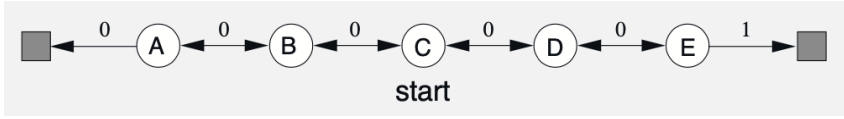
- TD(0)

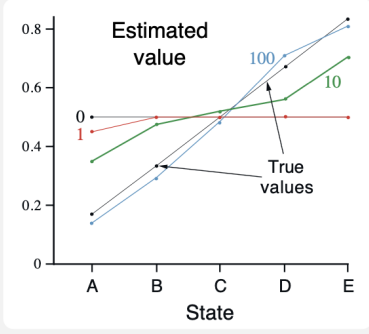
$$\widehat{V}^\pi(s_0) = \widehat{V}^\pi(s_0) + \alpha\delta_0.$$

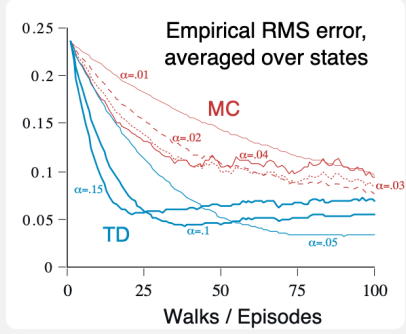
$\Rightarrow$ *Large bias* *["bootstrapping" on wrong values]*, *small variance*

Example: Random Walk

[1]







From Monte Carlo to Temporal Difference

Limitation of naive Monte-Carlo :

- performing a full trajectory is needed before the update
- we only update the value of the initial state s_0

Extension

- update the values of multiple states after each trajectory
- online updates, after each transition

Every Visit Monte Carlo (a.k.a. TD(1))

After the ith trajectory, instead of updating only s_0, for all $k = T_i - 1$ down to 0

$$\widehat{V}(s_{k,i}) = \widehat{V}(s_{k,i}) + \alpha_i(s_{k,i}) \left(\sum_{t=k}^{T_i} \gamma^{t-k} r_{t,i} - \widehat{V}(s_{k,i}) \right)$$

Remarks

- exact formula for episodic settings
- multiple updates of states visited more then once in the trajectory
- **first visit** variant: update $s_{k,i}$ only if $s_{k,i} \notin \{s_{0,i}, \ldots, s_{k-1,i}\}$

Every Visit Monte Carlo (a.k.a. TD(1))

After the ith trajectory, instead of updating only s_0, for all $k = T_i - 1$ down to 0

$$\widehat{V}(s_{k,i}) = \widehat{V}(s_{k,i}) + \alpha_i(s_{k,i}) \left(\sum_{t=k}^{T_i} \gamma^{t-k} \delta_{t,i} \right)$$

Remarks

- exact formula for episodic settings
- multiple updates of states visited more then once in the trajectory
- **first visit** variant: update $s_{k,i}$ only if $s_{k,i} \notin \{s_{0,i}, \ldots, s_{k-1,i}\}$

The $\mathcal{T}_\lambda^\pi$ Bellman operator

Definition

Given $\lambda < 1$, then the Bellman operator $\mathcal{T}_\lambda^\pi$ is

$$\mathcal{T}_\lambda^\pi = (1-\lambda) \sum_{m \geq 0} \lambda^m (\mathcal{T}^\pi)^{m+1}.$$

The $\mathcal{T}_\lambda^\pi$ Bellman operator

Definition

Given $\lambda < 1$, then the Bellman operator $\mathcal{T}_\lambda^\pi$ is

$$\mathcal{T}_\lambda^\pi = (1-\lambda) \sum_{m \geq 0} \lambda^m (\mathcal{T}^\pi)^{m+1}.$$

Remark: convex combination of the m-step Bellman operators $(\mathcal{T}^\pi)^m$ weighted by a sequences of coefficients defined as a function of a λ.

Temporal Difference $TD(\lambda)$

Idea: use the whole series of temporal differences to update $\widehat{V}^{\pi}$

- *Temporal difference* of a function $\widehat{V}^{\pi}$ for a transition $\langle s_t, r_t, s_{t+1} \rangle$

$$\delta_t = r_t + \gamma \widehat{V}^{\pi}(s_{t+1}) - \widehat{V}^{\pi}(s_t)$$

- Estimated value function

$$\widehat{V}^{\pi}(s_t) = \widehat{V}^{\pi}(s_t) + \alpha(s_t) \sum_{\tau=t}^{T} (\gamma\lambda)^{\tau-t} \delta_{\tau}$$

Temporal Difference $TD(\lambda)$

Idea: use the whole series of temporal differences to update $\widehat{V}^{\pi}$

- *Temporal difference* of a function $\widehat{V}^{\pi}$ for a transition $\langle s_t, r_t, s_{t+1} \rangle$

$$\delta_t = r_t + \gamma \widehat{V}^{\pi}(s_{t+1}) - \widehat{V}^{\pi}(s_t)$$

- Estimated value function

$$\widehat{V}^{\pi}(s_t) = \widehat{V}^{\pi}(s_t) + \alpha(s_t) \sum_{\tau=t}^{T} (\gamma\lambda)^{\tau-t} \delta_{\tau}$$

$\Rightarrow$ Still requires the whole trajectory before updating...

Temporal Difference $TD(\lambda)$: Eligibility Traces

- *Eligibility* traces $z \in \mathbb{R}^S$
- For every transition $s_t \rightarrow s_{t+1}$

Temporal Difference $TD(\lambda)$: Eligibility Traces

- *Eligibility* traces $z \in \mathbb{R}^S$
- For every transition $s_t \rightarrow s_{t+1}$
  1. Compute the temporal difference

$$\delta_t = r^\pi(s_t) + \gamma \widehat{V}^\pi(s_{t+1}) - \widehat{V}^\pi(s_t)$$

Temporal Difference $TD(\lambda)$: Eligibility Traces

- *Eligibility* traces $z \in \mathbb{R}^S$
- For every transition $s_t \rightarrow s_{t+1}$
  1. Compute the temporal difference

$$\delta_t = r^\pi(s_t) + \gamma \widehat{V}^\pi(s_{t+1}) - \widehat{V}^\pi(s_t)$$

  2. Update the eligibility traces

$$z(s) = \begin{cases} \lambda z(s) & \text{if } s \neq s_t \\ 1 + \lambda z(s) & \text{if } s = s_t \\ 0 & \text{if } s_t = s_0 \text{ (reset the traces)} \end{cases}$$

Temporal Difference $TD(\lambda)$: Eligibility Traces

- *Eligibility* traces $z \in \mathbb{R}^S$
- For every transition $s_t \rightarrow s_{t+1}$
  1. Compute the temporal difference

$$\delta_t = r^\pi(s_t) + \gamma \widehat{V}^\pi(s_{t+1}) - \widehat{V}^\pi(s_t)$$

  2. Update the eligibility traces

$$z(s) = \begin{cases} \lambda z(s) & \text{if } s \neq s_t \\ 1 + \lambda z(s) & \text{if } s = s_t \\ 0 & \text{if } s_t = s_0 \text{ (reset the traces)} \end{cases}$$

  3. For all state $s \in S$ *[all states are updated at each step]*

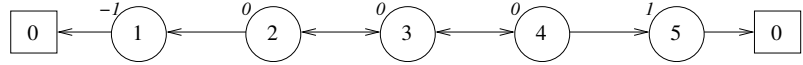
$$\widehat{V}^\pi(s) \leftarrow \widehat{V}^\pi(s) + \alpha(s) z(s) \delta_t.$$

Sensitivity to λ

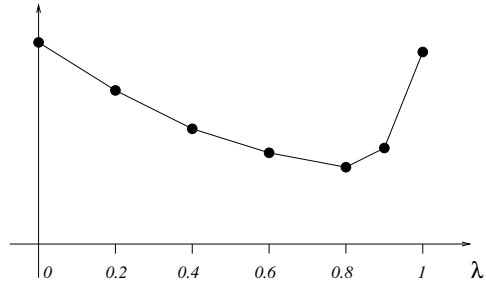
- $\lambda < 1$: *smaller variance* w.r.t. $\lambda = 1$ ($\approx$ inc-MC).
- $\lambda > 0$: *faster propagation* of rewards w.r.t. $\lambda = 0$.

Example: Sensitivity to λ

Linear chain example



Error $\sum_{s \in S} (\widehat{V}^{\pi}(s) - V^{\pi}(s))^2$ after $n = 100$ trajectories



Reinforcement Learning Algorithms

Tools

Policy Evaluation

Policy Learning

Policy Learning

Learn optimal policy π^*

For $i = 1, \ldots, n$

1. Set $t = 0$
2. Set initial state s_0
3. **While** ($s_{t,i}$ not terminal) *[execute one trajectory]*
   1. **Take action** $a_{t,i}$
   2. Observe next state $s_{t+1,i}$ and reward $r_{t,i} = r(s_{t,i}, a_{t,i})$
   3. Set $t = t + 1$

   EndWhile

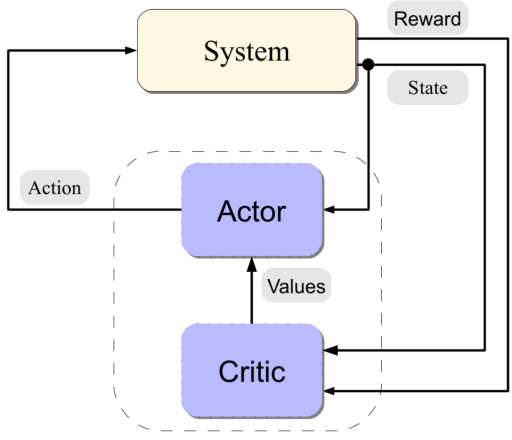
EndFor

Return: Estimate of the optimal policy $\widehat{\pi}^*$

Outline

Actor-Critic Methods

- Critic: estimate the value function of the policy given by the actor
- Actor: change the policy to improve the value given by the critic



Policy Iteration

1. Let π_0 be *any* stationary policy
2. At each iteration $k = 1, 2, \dots, K$
   - *Policy evaluation* given π_k, compute Q^{π_k}.
   - *Policy improvement*: compute the *greedy* policy

$$\pi_{k+1}(s) \in \arg\max_{a \in A} Q_k^{\pi}(s, a)$$

3. Return the last policy π_K

Policy Iteration is an instance of actor-critic methods:

- Actor: acts greedily (i.e., $\pi_{k+1} = \text{greedy}(Q^{\pi_k})$)
- Critic: compute Q given the policy π_k

Generalized Policy Iteration

More generally:

- Actor: *policy improvement* (even approximate)
- Critic: *policy evaluation*

[1] refers to actor/critic methods as **generalized policy iteration** (GPI)

👉 many possible schemas!

SARSA

Actor: the goal is to move the policy toward the greedy policy w.r.t. to the critic value

Possible strategies

- greedy (does not explore enough!)
- *ϵ-greedy* policy $\pi_Q(a|s)$

$$a = \mathcal{U}(\mathcal{A}) \qquad \text{with probability } \epsilon$$
$$a = \arg\max_{\overline{a}} Q(s, \overline{a}) \qquad \text{with probability } 1 - \epsilon$$

- *soft-max (random) exploratory* policy with temperature τ

$$\pi_Q(a|s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a'} \exp(Q(s,a')/\tau)}$$

The higher $Q(x, a)$, the more probability to take action a in state s

SARSA

Critic: uses TD to update the estimated Q-function

- Compute the temporal difference on the trajectory $\langle s_t, a_t, r_t, s_{t+1}, a_{t+1} \rangle$ (with actions chosen according to $\pi_Q(a|s)$)

$$\delta_t = r_t + \gamma \widehat{Q}(s_{t+1}, a_{t+1}) - \widehat{Q}(s_t, a_t)$$

SARSA

Critic: uses TD to update the estimated Q-function

- Compute the temporal difference on the trajectory $\langle s_t, a_t, r_t, s_{t+1}, a_{t+1} \rangle$ (with actions chosen according to $\pi_Q(a|s)$)

$$\delta_t = r_t + \gamma \widehat{Q}(s_{t+1}, a_{t+1}) - \widehat{Q}(s_t, a_t)$$

- Update the estimate of Q as

$$\widehat{Q}(s_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(s_t, a_t)\delta_t$$

State Action Reward State Action (SARSA) update

SARSA

Input: T, s_0, Q_0
$Q = Q_0$, $s = s_0$, $a = \mathcal{U}(\mathcal{A})$
$N = 0_{S \times A}$
for $t = 1, \ldots, T$ **do**
  N(s,a) = N(s,a) + 1
  Execute action a
  Observe r and next state s'
  $a' \sim \pi_Q(\cdot|s')$
  Update

$$\widehat{Q}(s,a) = \widehat{Q}(s,a) + \alpha_{N(s,a)}(r + \gamma \widehat{Q}(s', a') - \widehat{Q}(s,a))$$

  $s = s'$, $a = a'$
end
return $\widehat{Q}$

Examples

- $\pi_{t,Q}(a|s) = \dfrac{\exp(Q(s,a)/\tau_t}{\sum_b \exp(Q(s,b)/\tau_t)}$
- $\pi_{t,Q}(a|s) = (1-\epsilon_t)\mathbb{1}\left(a_t = \arg\max_b Q(s,b)\right) + \epsilon_t/A$

SARSA: Properties (informal)

- The TD updates make $\widehat{Q}$ converge to Q^{π}
- The update of π_Q allows to improve the policy
- A decreasing temperature allows to become more and more greedy
  $\Rightarrow$ If $\tau \to 0$ with a proper rate, then $\widehat{Q} \to Q^*$ and $\pi_Q \to \pi^*$
- Similarly for ϵ-greedy

SARSA: Limitations

The actions a_t need to be selected according to the current Q

$\Rightarrow$ **On-policy learning**

Outline

The Optimal Bellman Equation

Proposition

The optimal value function Q^* (i.e., $Q^* = \max_\pi Q^\pi$) is the solution to the optimal Bellman equation:

$$Q^*(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \max_{a' \in A} Q^*(s',a')\Big].$$

Q-Learning

Idea: use TD for the optimal Bellman operator

- Compute the (optimal) temporal difference on the trajectory $\langle s_t, a_t, r_t, s_{t+1} \rangle$ (with actions chosen **arbitrarily**!)

$$\delta_t = r_t + \gamma \max_{a'} \widehat{Q}(s_{t+1}, a') - \widehat{Q}(s_t, a_t)$$

- Update the estimate of Q as

$$\widehat{Q}(x_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(s_t, a_t)\delta_t$$

Q-Learning

Input: T, s_0, Q_0
$Q = Q_0$, $s = s_0$
$N = 0_{S \times A}$
for $t = 1, \ldots, T$ **do**
  Execute action $a \sim \pi_{t,Q}(s)$
  Observe r and next state s'
  N(s,a) = N(s,a) + 1
  Update

$$\widehat{Q}(s,a) = \widehat{Q}(s,a) + \alpha_{N(s,a)}(r + \gamma \max_{a'} \widehat{Q}(s', a') - \widehat{Q}(s,a))$$

  $s = s'$
end
return $\widehat{Q}$

Examples

- $\pi_{t,Q}(a|s) = \dfrac{\exp(Q(s,a)/\tau_t}{\sum_b \exp(Q(s,b)/\tau_t)}$
- $\pi_{t,Q}(a|s) = (1-\epsilon_t)\mathbb{1}\left(a_t = \arg\max_b Q(s,b)\right) + \epsilon_t/A$

Q-Learning: Properties

Proposition

If the learning rate satisfies the Robbins-Monro conditions in all states $s, a \in S \times A$

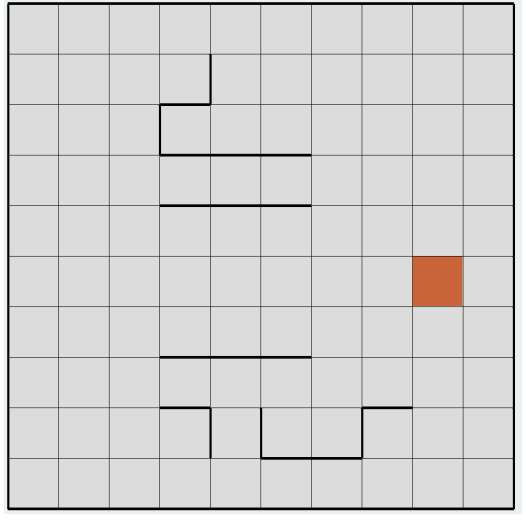
$$\sum_{i=0}^{\infty} \alpha_i(s,a) = \infty, \qquad \sum_{i=0}^{\infty} \alpha_i^2(s,a) < \infty,$$

and all state-action pairs are tried infinitely often, then for all $s, a \in S \times A$

$$\widehat{Q}(s,a) \xrightarrow{a.s.} Q^*(s,a)$$

Remark: this is another example of R-M algorithm
Remark: "infinitely often" requires a steady exploration policy

The Grid-World Problem



Q-Learning vs. SARSA

Update rule

- *Q-Learning*

$$\widehat{Q}(s_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(r_t + \gamma \max_{a'} \widehat{Q}(s_{t+1}, a') - \widehat{Q}(s_t, a_t))$$

- *SARSA*

$$\widehat{Q}(s_t, a_t) = \widehat{Q}(s_t, a_t) + \alpha(r_t + \gamma \widehat{Q}(s_{t+1}, a_{t+1}) - \widehat{Q}(s_t, a_t))$$

Both aim at learning the target policy $\pi^\star(s) = \arg\max_a Q^\star(s, a)$

- Q-learning can use any behavioral policy (*off-policy learning*)
- for SARSA the behavioral policy should be close to the estimated target policy (*on-policy learning*)

Summary

- Policy evaluation: Monte-Carlo and Temporal Difference (definition, pros and cons)
- Policy learning: SARSA and Q-learning (definition, guarantees)

Thank you!

facebook
Artificial Intelligence Research

Bibliography

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Bradford Book, MIT Press, Cambridge, MA, 1998.